CIFTS Workflows
IU Contribution

Joshua Hursey, Abhishek Kulkarni,
Timothy I. Mattox, Torsten Hoefler, Andrew Lumsdaine
Open Systems Laboratory
Indiana University
Bloomington, IN 47405
{jjhursey,adkulkar,timattox,htor,lums}@osl.iu.edu

February 20, 2009

Summary of Workflows

We believe that many of the workflows in the proposal are still applicable, but with some slight modifications. Most of the workflow discussion in this document focuses on the role of MPI (particularly Open MPI). Further iterations are needed to refine these workflows such that they are correct for other components of the CIFTS FTB.

Workflow: Node Failure

All of these workflows detail a response to a detected node failure.

- Section 1 Details the registered events for various components.
- Section 1.1 Node failure without a job
- Section 1.2 Node failure with MPI job aborting
- Section 1.3 Node failure with MPI job continuing

Workflow: Checkpoint/Restart & Process Migration

All of these workflows detail a response to a predicted node failure. So with advance notice of a failure, preventative actions are triggered to mitigate the impact of the failure. Additionally a RM/JS might wish to trigger a checkpoint to provide a coarse-grained, gang scheduling type of functionality.

- Section 2 Details the registered events for various components.
- Section 2.1 Gang Scheduling Support
- Section 2.2 Predicted node failure, resulting in a full job suspension/shutdown
- Section 2.3 Predicted node failure, resulting in process migration

Workflow: Interconnect Failure

All of these workflows detail a response to a faulty interconnect.

- Section 3 Details the registered events for various components.
- Section 3.1 Fail-over to an alternative device.
- Section 3.2 React to corrupted or missing data

Workflow: Task Farm

The task farm workflow concerns an MPI application that operates in a manager/worker model. This workflow still needs to be more concretely specified in a later draft.

1 Workflow: Node Failure

The following table details the events that each component will want to either throw or catch.

	Component	Action	Message
	Initialization & Job Launch		
0	RM/JS	Register	Check Problem Node (node *)
0	RM/JS	Register	Dead Physical Node (node *)
0	RM/JS	Register	Dead MPI Node (node *: job z)
0	RM/JS	Register	Restored Node (node *)
0	RM/JS	Register	Restored MPI Node (node *: job z)
0	Monitoring System	Register	Check Problem Node (node *)
0	Monitoring System	Register	Dead Physical Node (node *)
0	Monitoring System	Register	Restored Node (node *)
0	Autonomic Script	Register	Check Problem Node (node *)
0	Autonomic Script	Register	Dead Physical Node (node *)
0	MPI	Register	Dead MPI Node (node *: job z)
0	MPI	Register	Restored MPI Node (node *: job z)
0	MPI	Register	Dead MPI Rank (node x: job z: rank n-m)
0	Application	Register	Dead MPI Rank (node x: job z: rank n-m)
0	Application	Register	Restored MPI Node (node x: job z)

1.1 Workflow: Node Failure Without Job

A node failure can occur without any jobs running on the failed node.

	Component	**Action**	**Message**
	Node x Fails, no job running on node x		
1	Monitoring System	Throw	Check Problem Node (node x)
	Suspect problem with node x		
2(a)	RM/JS	Catch	Check Problem Node (node x)
	Suspend scheduling on node x (suspect failure)		
2(b)	Autonomic Script	Catch	Check Problem Node (node x)
	Attempt to confirm node x failed		
3	Autonomic Script	Throw	Dead Physical Node (node x)
	Confirmed node x failed		
4(a)	RM/JS	Catch	Dead Physical Node (node x)
	Remove node x from resource pool		
4(b)	Monitoring System	Catch	Dead Physical Node (node x)
	Remove node x from set of monitored resources		
4(c)	Autonomic Script	Catch	Dead Physical Node (node x)
	Notify sysadmin, trigger full diagnosis after hard reboot		
	Archives system logs, begin stress test, bring online spare nodes		
	Refund CPU accounting units, reschedule job		
	Time passes, machine returned to service		
5	Autonomic Script	Throw	Restored Node (node x)
	Sysadmin uses script to notify services of node recovery		
6(a)	RM/JS	Catch	Restored Node (node x)
	Return node x to resource pool		
6(b)	Monitoring System	Catch	Restored Node (node x)
	Return node x to the set of monitored resources		

1.2 Workflow: Node Failure With MPI Job Aborting

A node failure occurs while a job is running on the failed node. The policy expressed by the application through the MPI interface is that the MPI abort on such a failure.

	Component	**Action**	**Message**
	Node x Fails, job z running on allocation including node x		
1	Monitoring System	Throw	Check Problem Node (node x)
	Suspect problem with node x		
2(a)	RM/JS	Catch	Check Problem Node (node x)
	Mark node x as (suspect failure)		
2(b)	Autonomic Script	Catch	Check Problem Node (node x)
	Attempt to confirm node x failed		
3	Autonomic Script	Throw	Dead Physical Node (node x)
	Confirmed node x failed		
4(a)	RM/JS	Catch	Dead Physical Node (node x)
	Remove node x from resource pool		
4(b)	Monitoring System	Catch	Dead Physical Node (node x)
	Remove node x from set of monitored resources		
4(c)	Autonomic Script	Catch	Dead Physical Node (node x)
	Notify sysadmin, trigger full diagnosis after hard reboot		
	Archives system logs, begin stress test, bring online spare nodes		
	Refund CPU accounting units, reschedule job		
5	RM/JS	Throw	Dead MPI Node (node x: job z)
	Translates node x to job z		
6	MPI	Catch	Dead MPI Node (node x: job z)
	MPI prints console error, aborts job z		
	Time passes, machine returned to service		
7	Autonomic Script	Throw	Restored Node (node x)
	Sysadmin uses script to notify services of node recovery		
8(a)	RM/JS	Catch	Restored Node (node x)
	Return node x to unallocated resource pool		
8(b)	Monitoring System	Catch	Restored Node (node x)
	Return node x to the set of monitored resources		

1.3 Workflow: Node Failure With MPI Job Continuing

A node failure occurs while a job is running on the failed node. Node failure policy is that MPI should continue with holes in communicators. Node recovery policy is that MPI adds resources to internal pool to support application directed re-spawning of processes.

	Component	Action	Message
	Node x Fails, job z running on allocation including node x		
1	Monitoring System	Throw	Check Problem Node (node x)
	Suspect problem with node x		
2(a)	RM/JS	Catch	Check Problem Node (node x)
	Mark node x as (suspect failure)		
2(b)	Autonomic Script	Catch	Check Problem Node (node x)
	Attempt to confirm node x failed		
3	Autonomic Script	Throw	Dead Physical Node (node x)
	Confirmed node x failed		
4(a)	RM/JS	Catch	Dead Physical Node (node x)
	Remove node x from resource pool		
4(b)	Monitoring System	Catch	Dead Physical Node (node x)
	Remove node x from set of monitored resources		
4(c)	Autonomic Script	Catch	Dead Physical Node (node x)
	Notify sysadmin, trigger full diagnosis after hard reboot		
	Archives system logs, begin stress test, bring online spare nodes		
	Refund CPU accounting units, reschedule job		
5	RM/JS	Throw	Dead MPI Node (node x: job z)
	Translates node x to job z		
6	MPI	Catch	Dead MPI Node (node x: job z)
	Translate (node x:job z) to ranks m-n		
	Replace ranks m-n with MPI_PROC_NULL, call application error handlers		
7	MPI	Throw	Dead MPI Rank (node x: job z: rank n-m)
	Translate (node x:job z) to ranks m-n		
8	Application	Catch	Dead MPI Rank (node x: job z: rank n-m)
	Work around 'blank' ranks n-m in the MPI communicators		
	Time passes, machine returned to service		
9	Autonomic Script	Throw	Restored Node (node x)
	Sysadmin uses script to notify services of node recovery		
10(a)	RM/JS	Catch	Restored Node (node x)
	Return node x to resource pool for job z		
10(b)	Monitoring System	Catch	Restored Node (node x)
	Return node x to the set of monitored resources		
11	RM/JS	Throw	Restored MPI Node (node x: job z)
	Translates node x to job z		
12(a)	MPI	Catch	Restored MPI Node (node x: job z)
	Add node x as an unallocated resource		
12(b)	Application	Catch	Restored MPI Node (node x: job z)
	If needed, use MPI_Comm_spawn to create new processes		

2 Workflow: Checkpoint/Restart & Process Migration

All of these workflows detail a response to a predicted node failure. So with advance notice of a failure, preventative actions are triggered to mitigate the impact of the failure. Additionally a RM/JS might wish to trigger a checkpoint to provide a coarse-grained, gang scheduling type of functionality.

	Component	**Action**	**Message**
	Initialization & Job Launch		
0	RM/JS	Register	Restored Node (node *)
0	RM/JS	Register	Suspend Job (job z)
0	RM/JS	Register	Resume Job (job z)
0	RM/JS	Register	Resume Job Cmd (job z)
0	RM/JS	Register	Predict Problem Node (node *)
0	RM/JS	Register	Migrate Node (job z: node x,q)
0	RM/JS	Register	Migrate Node Done (job z: node x,q)
0	Autonomic Script	Register	Restored Node (node *)
0	Autonomic Script	Register	Predict Problem Node (node *)
0	MPI	Register	Suspend Job (job z)
0	MPI	Register	Resume Job (job z)
0	MPI	Register	Resume Job Cmd (job z)
0	MPI	Register	Migrate Node (job z: node x,q)
0	MPI	Register	Migrate Node Done (job z: node x,q)

2.1 Workflow: Gang Scheduling Support

Gang scheduling support. The RM/JS suspends and resumes entire jobs using a checkpoint/restart technique in cooperation with the MPI implementation.

	Component	**Action**	**Message**
	RM/JS decides to suspend job z using CPR		
1	RM/JS	Throw	Suspend Job (job z)
	Suspend job z		
2	MPI	Catch	Suspend Job (job z)
	Coordinate a global checkpoint operation. Suspend/Terminate job z		
3	MPI	Throw	Resume Job Cmd (job z)
	Provide RM/JS with the command needed to resume job z		
4	RM/JS	Catch	Resume Job Cmd (job z)
	Store command with information for job z		
	RM/JS decides to resume job z from CPR		
5	RM/JS	Throw	Resume Job (job z)
	Use stored resume information for job z to restart job		
6	MPI	Catch	Resume Job (job z)
	Bring job z back into a running state		

2.2 Workflow: Predicted Failure, Job Suspend

A monitoring system predicts a node failure based on heuristic information gathered from the operating system, network card, and other system resources. The job is suspended and rescheduled for later execution.

	Component	**Action**	**Message**
	RM/JS decides to suspend job z using CPR		
1	Autonomic Script	Throw	Predict Problem Node (node x)
	Information gathered indicates emanate failure of node x		
2	RM/JS	Catch	Predict Problem Node (node x)
	Suspend scheduling on node x (predicted failure)		
	Translate node x to job z		
3	RM/JS	Throw	Suspend Job (job z)
	Suspend job z		
4	MPI	Catch	Suspend Job (job z)
	Coordinate a global checkpoint operation. Suspend/Terminate job z		
5	MPI	Throw	Resume Job Cmd (job z)
	Provide RM/JS with the command needed to resume job z		
6	RM/JS	Catch	Resume Job Cmd (job z)
	Store command with information for job z		
	Reschedule job z		
	Job z becomes runnable once again		
7	RM/JS	Throw	Resume Job (job z)
	Use stored resume information for job z to restart job		
8	MPI	Catch	Resume Job (job z)
	Bring job z back into a running state		
	Time passes, node x returned to service		
9	Autonomic Script	Throw	Restored Node (node x)
	Information gathered indicates node x is stable again		
10	RM/JS	Catch	Restored Node (node x)
	Return node x to resource pool		

2.3 Workflow: Predicted Failure, Process Migration

A monitoring system predicts a node failure based on heuristic information gathered from the operating system, network card, and other system resources. The job is suspended and rescheduled for later execution.

	Component	**Action**	**Message**
	RM/JS decides to suspend job z using CPR		
1	Autonomic Script	Throw	Predict Problem Node (node x)
	Information gathered indicates emanate failure of node x		
2	RM/JS	Catch	Predict Problem Node (node x)
	Suspend scheduling on node x (predicted failure)		
	Translate node x to job z		
3	RM/JS	Throw	Migrate Node (job z: node x,q)
	Allocate spare node q to job z		
	Migrate processes from job z on node x to new node q		
4	MPI	Catch	Migrate Node (job z: node x,q)
	Coordinate a global checkpoint operation.		
	Migrate ranks from node x to new node q. Resume application		
5	MPI	Throw	Migrate Node Done (job z: node x,q)
	Tell RM/JS that migration is finished		
6	RM/JS	Catch	Migrate Node Done (job z: node x,q)
	Receive confirmation that node x no longer contains MPI ranks		
	Time passes, node x returned to service		
7	Autonomic Script	Throw	Restored Node (node x)
	Information gathered indicates node x is stable again		
8	RM/JS	Catch	Restored Node (node x)
	Return node x to resource pool		

3 Workflow: Faulty Interconnect

The following table details the events that each component will want to either throw or catch.

	Component	**Action**	**Message**
	Initialization & Job Launch		
0	RM/JS	Register	Failed Physical Interface (iface *: node *)
0	RM/JS	Register	Failed MPI Physical Interface (iface *: node *: job *)
0	RM/JS	Register	Restored Physical Interface (iface *: node *)
0	RM/JS	Register	Restored MPI Physical Interface (iface *: node *: job *)
0	RM/JS	Register	MPI Message Corruption (node *: job *)
0	IB Fault Monitor	Register	Failed Physical Interface (iface *: node *)
0	IB Fault Monitor	Register	Restored Physical Interface (iface *: node *)
0	IB Fault Monitor	Register	Check Physical Interface (iface *: node *)
0	Autonomic Script	Register	Failed Physical Interface (iface *: node *)
0	Autonomic Script	Register	Restored Physical Interface (iface *: node *)
0	Autonomic Script	Register	Check Physical Interface (iface *: node *)
0	MPI	Register	Failed MPI Physical Interface (iface *: node *: job z)
0	MPI	Register	Restored MPI Physical Interface (iface *: node *: job z)
0	MPI	Register	MPI Message Corruption (node *: job z)

3.1 Workflow: Fail-over to an Alternative Device

A physical network interface fails, MPI fails-over to an alternative device and continues.

	Component	**Action**	**Message**
	Interface p fails on node x, job z running on node x		
	IB Fault Monitor is first to detect		
1	IB Fault Monitor	Throw	Failed Physical Interface (iface p: node x)
	Interface p on node x has failed		
2(a)	RM/JS	Catch	Failed Physical Interface (iface p: node x)
	Translate node x to job z		
2(b)	Autonomic Script	Catch	Failed Physical Interface (iface p: node x)
	Attempt diagnose and clean up IB routes and switches		
3	RM/JS	Throw	Failed MPI Physical Interface (iface p: node x: job z)
	Notify MPI of failed interface		
4	MPI	Catch	Failed MPI Physical Interface (iface p: node x: job z)
	Mark interface p as down		
	If possible, use an alternative interface		
	If not, suspend communication until interface restored		
	Interface p returned to service on node x		
5	Autonomic Script	Throw	Restored Physical Interface (iface p: node x)
	Interface p has been restored to service on node x		
6(a)	IB Fault Monitor	Catch	Restored Physical Interface (iface p: node x)
	Confirm interface is restored		
6(b)	RM/JS	Catch	Restored Physical Interface (iface p: node x)
	Translate node x to job z		
7	RM/JS	Throw	Restored MPI Physical Interface (iface p: node x: job z)
	Notify MPI of restored/new interface p		
8	MPI	Catch	Restored MPI Physical Interface (iface p: node x: job z)
	Add interface p back to the possible interfaces for communication		

3.2 Workflow: React to Corrupted or Missing Data

A physical network interface is dropping or corrupting packets. MPI takes corrective action to mask such fails. At some point MPI may decide to remove the interface from service similar to Section 3.1

	Component	**Action**	**Message**
	Interface p dropping or corrupting packets on node x		
	MPI is first to detect		
1	MPI	Throw	MPI Message Corruption (node x: job z)
	MPI detects message corruption		
	Continue masking corruption while interfaces are inspected		
2	RM/JS	Catch	MPI Message Corruption (node x: job z)
	Translate node x to iface p-q		
3	RM/JS	Throw	Check Interface (iface p-q: node x)
	Ask script to check interfaces for suspected failure		
4(a)	Autonomic Script	Catch	Check Interface (iface p-q: node x)
	Checks interfaces		
4(b)	IB Fault Monitor	Catch	Check Interface (iface p-q: node x)
	Checks interfaces		
5	Autonomic Script	Throw	Failed Physical Interface (iface p: node x)
	Notify of confirmed failed interface		
6	RM/JS	Catch	Failed Physical Interface (iface p: node x)
	Translate node x to job z		
7	RM/JS	Throw	Failed MPI Physical Interface (iface p: node x: job z)
	Notify MPI of failed interface		
8	MPI	Catch	Failed MPI Physical Interface (iface p: node x: job z)
	Mark interface p as down		
	If possible, use an alternative interface		
	If not, suspend communication until interface restored		
	Interface p returned to service on node x		
9	Autonomic Script	Throw	Restored Physical Interface (iface p: node x)
	Interface p has been restored to service on node x		
10(a)	IB Fault Monitor	Catch	Restored Physical Interface (iface p: node x)
	Confirm interface is restored		
10(b)	RM/JS	Catch	Restored Physical Interface (iface p: node x)
	Translate node x to job z		
11	RM/JS	Throw	Restored MPI Physical Interface (iface p: node x: job z)
	Notify MPI of restored/new interface p		
12	MPI	Catch	Restored MPI Physical Interface (iface p: node x: job z)
	Add interface p back to the possible interfaces for communication		